University of Pittsburgh

Algorithms and Data Structures 2
CS 1501

Spring 2023

Sherif Khattab

ksm73@pitt.edu

(Slides are adapted from Dr. Ramirez's and Dr. Farnan's CS1501 slides.)

Announcements

- Upcoming Deadlines
  - Homework 4: this Friday @ 11:59 pm
  - Lab 3: Tuesday 2/14 @ 11:59 pm
  - Assignment 1: Friday 2/17 @ 11:59 pm
- Please make your Piazza posts public as much as possible

Previous lecture

- Red-Black BST (self-balancing BST)
  - add
  - delete
  - runtime of operations
- Turning recursive tree traversals to iterative

This Lecture

- Binary Search Tree uses comparisons between keys to guide the searching
- What if we use the digital representation of keys for searching instead?
  - Keys are represented as a sequence of digits (e.g., bits) or alphabetic characters
- Digital Searching Problem

Digital Searching Problem

- Input:
  - a (large) dynamic set of data items in the form of
    - *n* (key, value) pairs; key is a string from an alphabet of size *R*
    - Each key has *b* bits or *w* characters (the chars are from the alphabet)
    - What is the relationship between *b* and *w*?
  - a *target key (k)*
- Output:
  - The corresponding value to *k* if target key found
  - Key not found otherwise

Digital Search Trees (DSTs)

Instead of looking at less than/greater than, lets go left or right based on the bits of the key

So, we again have 4 options:

- current node is null, k not found
- k is equal to the current node's key, k is found, return corresponding value
- current bit of k is 0, continue to left child
- current bit of k is 1, continue to right child

DST example: Insert and Search

Insert:

4	0100
3	0011
2	0010
6	0110
5	0101

Search:

3	0011
7	0111